Framework for path finding in multi-layer transport networks

Dijkstra, F.

Publication date
2009
Document Version
Final published version

Chapter 5

Network Description Language

This chapter is based on *Using RDF to Describe Networks* by J.J. van der Ham, F. Dijkstra, F. Travostino, H.M.A. Andree and C.T.A.M. de Laat [a3] and on *A Distributed Topology Information System for Optical Networks Based on the Semantic Web* by J.J. van der Ham, F. Dijkstra, P. Grosso and R. van der Pol, A. Toonk and C.T.A.M. de Laat [a9]. This chapter would not have been possible without the kind permission of Jeroen van der Ham to include part of his work here.

5.1 Introduction

The routing step, which is required for provisioning of circuit switched network connections, is responsible for distributing topology information and network state across different domains. This chapter examines the distribution of topology information. It presents the Network Description Language (NDL), which builds upon the Resource Description Format (RDF) [u12] and its linking capabilities to produce a distributed Topology Knowledge Base (TKB) [p41]. It is worth to emphasise that proposed network description language is only a method to describe topology information. It does not enforce a specific way of distributing this information, nor does it eliminate the need for a control plane for signalling and provisioning.

The modelling process, as outlined in figure 4.2 in the previous chapter 4, consists of two steps, the mapping from network to model and the mapping from model to syntax. In chapters 2 and 4 took the first step. Chapter 2 defined the terminology and model for transport networks and chapter 4 defined a model for multi-layer networks. In this chapter and the following chapter we

take the second step. This chapter defines a syntax for describing (a single layer of a) transport network, while the next chapter defines a syntax for describing multi-layer networks.

5.2 Introduction to the Semantic Web

The World Wide Web has allowed us to publish and share documents and information with other people in the world. However, because the web has become so popular and so widespread, it has almost become the victim of its own success. Because of the large-scale and the abundant availability of data, it becomes very hard to find what we want. Search-machines, such as Google or Yahoo, have come to the rescue and have indexed the data. However, computers still have no common sense, so the search capabilities of the search machines are rather limited. Consider for example the following two sentences:

- A is connected to B.
- There is a connection between A and B.

Even humans can differ in opinion whether these two sentences have the same meaning. So there is no way that a computer without common sense will understand that these two lines mean the same thing. This is where the Semantic Web comes to the aid of computers (and people). The following is an excerpt of the activity statement of the Semantic Web initiative [u14]:

> *The goal of the Semantic Web initiative is to create a universal medium for the exchange of data where data can be shared and processed by automated tools as well as by people. For the Web to scale, tomorrow's programs must be able to share and process data even when these programs have been designed totally independently.*

In 2000 the Semantic Web initiative was started by the World Wide Web Consortium (W3C). Since then they have been working on several specifications to publish and share (meta)data, including the Resource Description Framework (RDF) [u12]. In the following section we provide a brief introduction to RDF. Examples of RDF are given later on in section 5.3.

5.2.1 Resource Description Framework

The Resource Description Framework (RDF) is a method for representing information about resources on the Web. It provides a common framework for

expressing metadata so that it can be exchanged between applications without loss of meaning.

Information in RDF is expressed as statements. Each statement is a triplet, with the following elements:

Subject The resource being described

Predicate The property of the subject that is described

Object The value of the property for the subject

A set of triplets is called a graph. Using the property that an object can also be the subject of another triplet, complex graphs can be created. An example of such a graph is shown in figure 5.1. This graph shows that `Thesis` is written by `Freek Dijkstra` and it also provides some additional information about him, his name, affiliation and email address.

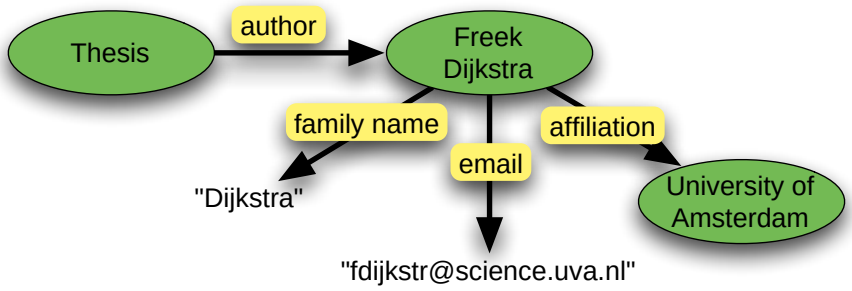

Figure 5.1: *A simple RDF graph (source: Jeroen van der Ham)*

The graph shown in figure 5.1 still has the same problem as the two lines as shown before; we have provided an abstract way of defining relations, but we still use plain English as labels for identifying these relations. Consider for example the `author` relationship, we could also have expressed this as `creator` without much loss of meaning to human readers. RDF solves this terminology problem by using Uniform Resource Identifiers (URIs). Related terms are usually defined using the same URI-prefix, taking the form of XML namespaces. See for example the Dublin Core Metadata Initiative [s58].

5.2.2 RDF Schemata

```
1 <http://www.science.uva.nl/~fdijkstr/thesis/> <http://purl.org/dc/elements/1.1/creator>
      <http://www.macfreek.nl/freek/#freek>.
2 <http://www.macfreek.nl/freek/#freek> <http://xmlns.com/foaf/0.1/family_name> "Dijkstra".
3 <http://www.macfreek.nl/freek/#freek> <http://xmlns.com/foaf/0.1/mbox>
      "fdijkstr@science.uva.nl".
4 <http://www.macfreek.nl/freek/#freek>
      <http://www.w3.org/2002/01/p3prdfv1#thirdparty.employer> <http://www.uva.nl/>.
```

Listing 5.1: *The N3 representation of the semantic graph of figure 5.1.*

Listing 5.1 describes the semantic graph of figure 5.1. Each triplet contains three URIs followed by a dot to indicate the end of a triplet. The notation used in this listing is N-triplet, which is a subset of the Notation3 syntax [t5].

The predicates defined in this list are URIs with a well-defined meaning. For example, the `author` relationship is defined by the URI `http://purl.org/dc/elements/1.1/creator`, which is defined by the Dublin Core Metadata Initiative [s58] in their Metadata Element Set. Related terms are defined in the same XML namespaces with the URI-prefix `http://purl.org/dc/elements/1.1/`.

```
1 <http://www.macfreek.nl/freek/#freek> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
      <http://xmlns.com/foaf/0.1/Person>.
```

Listing 5.2: *Use of the rdf:type predicate.*

An XML namespace with definitions of related terms is called an RDF schema. RDF schemata define the URIs and properties of RDF classes and RDF predicates. RDF classes define the types of subjects and objects. Listing 5.2 defines `Freek Dijkstra` to be (an instance of) a person. RDF predicates define the properties of attributes of instances. For example the Dublin Core definition for `creator` gives a name, description, type and version information. A predicate definition can also contain the compulsory classes for the subject (the domain of a predicate) and object (the range of a predicate).

It is possible to state triplets, but only if these statements use predicated and classes defined in RDF schemata, a computer can reason about its meaning. For example, a computer can find other documents created by the same author, or reason that the range of an author is a human being, and the author of this thesis is Freek Dijkstra, then Freek Dijkstra must be a human being.

5.2.3 RDF versus XML

There are several ways of expressing RDF graphs, one is the graphical form as in figure 5.1, and another is the statements of triplets in Notation3 in listing 5.1. The most common textual form is *RDF/XML* [s55], where the graph is encoded in an XML format. Throughout this thesis we use the RDF/XML notation, which allows us to leverage tools for XML as well as RDF. Examples and explanation of the RDF/XML syntax are given in the next section.

Besides that RDF allows *reasoning* about statements, it also has a few other technical advantages over other descriptions languages, such as plain XML.

Unique Identification Objects in RDF are identified by a URI. This is an advantage in multi-domain environments, since it makes it easy to clearly and uniquely define network elements in requests.

Flexible Graph Structure The relations between network elements can lead to cycles in the relation-graph. RDF extends the tree structure of XML with reference pointers so that it is able to deal with cycles.

Distributed Descriptions In order to describe inter-domain connections, the interrelation of different (operational) network domains must be described. Each domain must be able to independently publish its own network information and point to other network domains. The RDF `seeAlso` predicate provides an elegant solution for this problem. We will get back to this in section 5.3.3.

Extendable RDF schemata are easily extensible. That is, it allows users to publish all information they care about, and mix it with network schemata. The extensibility applies to both current schemata (e.g. geographic information or organisational information in `geo` and `vcard`), as well as future schemata.

5.3 Network Description Language

Given that we want to describe extensible, distributed network descriptions, we set out to create a simple ontology in RDF to describe networks. The result of this work is the Network Description Language (NDL).

5.3.1 Topology Schema

The Network Description Language consists of multiple schemata, each describing a separate part of the ontology.

The topology schema of the network description language is the ontology we created to describe the topology of computer networks. An overview of the classes and properties of the topology schema is given in figure 5.2.

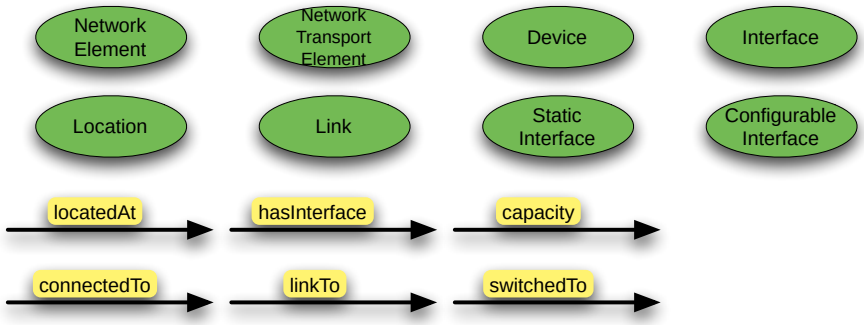

Figure 5.2: *Overview of the classes and predicates in the NDL topology schema*

NDL has eight classes, shown at the top, that define what kind of resources can be described. The three main classes are:

Location Physical places where devices are located.

Device Devices that are part of a network.

Interface The interfaces with which devices are connected to a network.

NDL has six properties, shown at the bottom in the figure, to define the relations between instances of the classes and other information.

locatedAt A relation between resources and their location.

hasInterface A relation between devices and interfaces.

linkTo A relation between two interfaces, describing that they are *externally* connected with a direct link.

connectedTo This property is similar to *linkTo*, but the connection does not have to be direct; the interfaces may be connected by a series of links.

switchedTo This property is used to describe cross connects, *internal* connections within a device.

In addition, the predicates *label* from the RDF schema, and *description* from the Dublin Core schema can be used to describe the name and description of network elements.

A more extensive definition of the different classes and predicates can be found in the NDL schemata itself [u3].

5.3.2 Domain Schema

The topology schema allows the description of physical network topologies. The NDL domain schema also allows group description of devices, links and interfaces in networks. The classes and predicates of the domain schema are shown in figure 5.3.

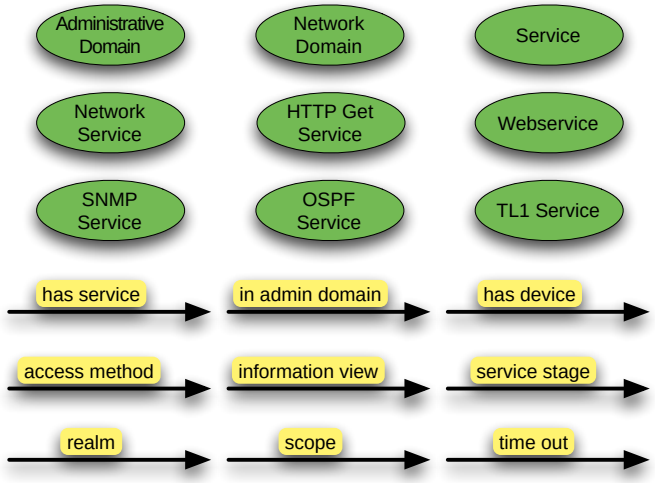

Figure 5.3: *Overview of the classes and predicates in the Network Description Language domain schema*

The two main classes in the domain schema are:

NetworkDomain is a collection of network elements. It behaves very similar to a Devices in the topology schema, but describes a domain rather than a physical device.

AdministrativeDomain is a organisational entity that is responsible for the operational control of resources (including network resources).

While a network domain is focused on the physical network, the operational domain is focused on the organisation of the control plane. An operator domain as defined in section 2.4.3 translates to a NetworkDomain (a collection of network elements with the same operator). The operator as defined in section 2.4.3 translates to the AdministrativeDomain[1].

Using the combination of the topology and domain schema, it is possible to create descriptions of network domains. An example of such a description is shown in listing 5.3. The picture in figure 5.4 shows what is being described.

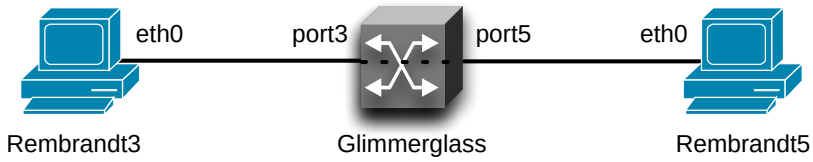

Figure 5.4: *A simple network.*

Lines 14 to 18 of listing 5.3 define the device `Rembrandt3`. The `#`-prefix on line 14 states that the device is defined in the local namespace. Line 15 provides a human readable name and line 16 states that this device is located in the location `Lighthouse` (defined on lines 11 to 13). Finally, line 17 defines that `Rembrandt3` has an interface, `Rembrandt3:eth0`. This interface is defined on lines 19 to 22. The connection to another interface is defined using the `linkTo` property on line 21, in this case it is defined to be connected to `Glimmerglass:port3`. The `Glimmerglass` device is defined similarly on lines 23–38, and the `Rembrandt5` device on lines 39–47.

The connection between the `Rembrandt3` and the `Glimmerglass` is defined in both directions. This is used to denote a duplex connection and further ensures the consistency of the description.

Our network description does not only contain a topology description, but also describes the current configuration of the Glimmerglass device. The `switchedTo` statement in line 32 states that the `Glimmerglass:port3` has an *internal* connection to `Glimmerglass:port5`.

Just like the `linkTo` property, the `switchedTo` property must be defined in both directions. So the inverse `switchedTo` property from `Glimmerglass:port5` to `Glimmerglass:port3` is also be given on line 37. With the `linkTo` and `switchedTo` statements as given above, we have defined a path from the device `Rembrandt3` to `Rembrandt5`.

[1] These class names resemblances an older terminology we used.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ndl="http://www.science.uva.nl/research/sne/ndl#"
         xmlns:domain="http://www.science.uva.nl/research/sne/ndl/domain#">
    <domain:NetworkDomain rdf:about="#UvALight">
        <rdf:label>UvA Light</rdf:label>
        <domain:hasDevice rdf:resource="#Rembrandt3"/>
        <domain:hasDevice rdf:resource="#Rembrandt5"/>
        <domain:hasDevice rdf:resource="#Glimmerglass"/>
    </domain:NetworkDomain>
    <ndl:Location rdf:about="#Lighthouse">
        <rdf:label>Lighthouse</rdf:label>
    </ndl:Location>
    <ndl:Device rdf:about="#Rembrandt3">
        <rdf:label>Rembrandt3</rdf:label>
        <ndl:locatedAt rdf:resource="#Lighthouse"/>
        <ndl:hasInterface rdf:resource="#Rembrandt3:eth0"/>
    </ndl:Device>
    <ndl:Interface rdf:about="#Rembrandt3:eth0">
        <rdf:label>eth0</rdf:label>
        <ndl:linkTo rdf:resource="#Glimmerglass:port3"/>
    </ndl:Interface>
    <ndl:Device rdf:about="#Glimmerglass">
        <rdf:label>Glimmerglass</rdf:label>
        <ndl:locatedAt rdf:resource="#Lighthouse"/>
        <ndl:hasInterface rdf:resource="#Glimmerglass:port3"/>
        <ndl:hasInterface rdf:resource="#Glimmerglass:port5"/>
    </ndl:Device>
    <ndl:Interface rdf:about="#Glimmerglass:port3">
        <rdf:label>port3</rdf:label>
        <ndl:linkTo rdf:resource="#Rembrandt3:eth0"/>
        <ndl:switchedTo rdf:resource="#Glimmerglass:port5"/>
    </ndl:Interface>
    <ndl:Interface rdf:about="#Glimmerglass:port5">
        <rdf:label>port5</rdf:label>
        <ndl:linkTo rdf:resource="#Rembrandt5:eth0"/>
        <ndl:switchedTo rdf:resource="#Glimmerglass:port3"/>
    </ndl:Interface>
    <ndl:Device rdf:about="#Rembrandt5">
        <rdf:label>Rembrandt5</rdf:label>
        <ndl:locatedAt rdf:resource="#Lighthouse"/>
        <ndl:hasInterface rdf:resource="#Rembrandt5:eth0"/>
    </ndl:Device>
    <ndl:Interface rdf:about="#Rembrandt5:eth0">
        <rdf:label>eth0</rdf:label>
        <ndl:linkTo rdf:resource="#Glimmerglass:port5"/>
    </ndl:Interface>
</rdf:RDF>
```

Listing 5.3: *An example description of the network of figure 5.4.*

5.3.3 Distributed Repositories

So far we have described how to create descriptions of (local) networks and how to gather information from these descriptions.

In multi-domain environments there is a big potential for inconsistencies if information for each domain is not centrally maintained, and each domain replicates network descriptions of other domains. NDL addresses this issue by creating a distributed topology description, where descriptions link to each other. These links are provided using RDF's `seeAlso` statement, which points to other documents. An example of this is shown in listing 5.4.

```
1 <ndl:Interface rdf:about="#Glimmerglass:port27">
2     <ndl:name>Glimmerglass:port27</ndl:name>
3     <ndl:linkTo rdf:resource="http://www.netherlight.nl/ndl.rdf#C6509:port7"/>
4 </ndl:Interface>
5 <!-- test -->
6 <ndl:Interface rdf:about="http://www.netherlight.nl/ndl.rdf#C6509:port7">
7     <rdfs:seeAlso rdf:resource="http://www.netherlight.nl/ndl.rdf"/>
8 </ndl:Interface>
```

Listing 5.4: *Example of distributed repositories.*

As shown before, lines 1 to 4 show the description of an interface of the `Glimmerglass`. However, note that in line 3, this port is defined to be connected to `http://www.netherlight.nl/ndl.rdf#C6509:port7`. On lines 5 to 7 is the definition of the interface `http://www.netherlight.nl/ndl.rdf#C6509:port7`. The `rdfs:seeAlso` statement is used to link to the network description of NetherLight. An application or crawler can then follow this pointer to the description of NetherLight and get more information about the interface there.

Concluding, it is possible to create a distributed network description, without a central repository.

5.3.4 Addressing

RDF uses Uniform Resource Identifiers (URIs) [s21] to denote classes, properties, and instances of classes. For example, each Device or Interfaces is identified using a URI. This provides an elegant solution to make sure each object is unique, since only the owner of a certain domain may publish authoritative information of objects with an URI in his namespace.

In addition, the use of URIs along with explicit properties for each object follows the separation of naming and addressing [t14, t7]. This is an important

concept, since the name of an object does not change, even if its properties do change.

Concluding, the use of semantic web provides the URI as a solution for globally unique addressing of network resources. . The essential advantage is that information is kept only at the owner of the resource.

5.3.5 Extensibility

The use of RDF allows easy extension of the topology and domain schema with other schemata. In chapter 6 we will see extensions to layer and capability descriptions.

Our vision is that a application will be able to consume the descriptions of all the architectural components that form an end-to-end infrastructure. This information include computing resources, storage resources, visualisation resources, network resources, content descriptions, etc. etc. All resources can be linked with loose couplings to allow a metascheduler application to orchestrate all resources together in a combined effort [a6].

5.4 Applications

NDL provides a powerful language to solve many of the operational issues that operators and users face in hybrid networks. It allows the automatic creation of network maps; it facilitates path finding algorithms used by reservation and network management systems; it enhances the interoperability and the exchange of information between different operational domains.

This section highlights a few of these applications [p15, a9].

5.4.1 Visualisation using RDF tools

Network maps are a visual aid used by network engineers to design, examine and troubleshoot circuits.

One of the advantages of using NDL as the language for description of hybrid networks is the availability of semantic web tools for RDF, which can parse and consume the information in each NDL file. This means that extracting the information needed for network management is straightforward and simple.

Van der Ham implemented a visualisation tool that takes a network description in NDL format, and uses a SPARQL query [s56] to get the connections between the devices and their names. Using a small script, this data is then

converted to serve as input to GraphViz, an open source graph visualisation tool [u9].

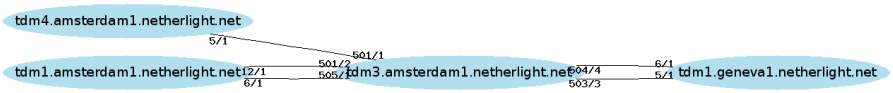

Figure 5.5: *A graph of NetherLight resources, as extracted from a NDL file (Source: Van der Ham [p15])*

An example of such a graph is shown in figure 5.5. This is the map of NetherLight [u11], one of the network domains participating in the GLIF. The script used to generate the graph can be found at the NDL homepage [u3].

5.4.2 Path Finding and Google Mash-up

At SuperComputing 2006, Van der Ham *et al.* demonstrated path finding in the GLIF infrastructure [p14]. This infrastructure consists of interconnected optical exchanges. For this demonstration most of the GOLEs provided a description of their network in NDL format. A key feature is that each description was published independently of the others, allowing GOLEs to stay in charge of the data they publish. The descriptions defined the physical resources in the GOLEs and the links to other GOLEs.

An application gathered all the NDL files from the different GOLEs, and generates a list of endpoints. Using a web interface, a user can select two endpoints from this list, and the application applies the Dijkstra algorithm [p11] to find the shortest path between the two endpoints. The resulting path is displayed in the web browser as a highlighted path through the network graphically presented using Google Maps. A list of hops is also provided next to the map. Figure 5.6 shows the example output for a path between Seattle and Geneva.

This application is mostly a proof of concept, because there are still more challenges with regard to inter-domain path finding. Issues such as policy and multi-layer network need to be addressed, as well as information regarding utilisation. The NDL domain schema can help here by providing pointers to relevant information services on policy, utilisation, or reservations. This is part of future implementations.

Figure 5.6: *Path finding in GLIF, presented in Google Maps.*

5.4.3 Lightpath Planning in SURFnet6

SURFnet6 is the Dutch national research and education network. SURFnet6 is a hybrid network, offering both IP services and lightpath services. Toonk and Van der Pol of the Dutch national supercomputing centre SARA have written a tool for planning new lightpaths based on NDL [p30, p31]. In this application they automatically generate an NDL topology description of the SURFnet6 network based on information gathered from network devices using their TL1-Toolkit [u7]. Additionally, a network state database holds the cross-connect information for each network element in the network. That is, information about currently provisioned lightpaths. This enables the application to determine the amount of time-slots still available on each interface.

A network operator at the SURFnet 6 network operations centre (NOC) can use a web interface to query for a lightpath between two endpoints of the SURFnet6 network. The user first selects two endpoints from a list of the available endpoints and specifies some properties for this new lightpath, such as a name, the capacity and whether this should be a protected or unprotected path. Subsequently, a software tool uses the information from the NDL file to construct a graph of the network. Using the network state database, this graph is pruned taking out sections that do not have enough bandwidth available. Then the Dijkstra algorithm is applied using the current load of the network

as link metric.

5.4.4 Lightpath Monitoring in NetherLight

NDL can play an important role in lightpath monitoring as well. SARA developed Spotlight, a tool for lightpath monitoring in SURFnet 6 and in NetherLight.

To monitor the lightpaths, SARA uses NDL to specify their topology details, and actively query the network elements involved. The output is stored in a network state database with alarm and configuration information. This enables us to correlate the configuration data with the alarm information and determine whether a specific lightpath is up or down. If a failure is detected somewhere in the lightpath route, this will be clearly indicated using a visualisation of the lightpath. The Spotlight application is available online, see [u5].

5.5 Conclusion

In this chapter we have introduced a way of applying the Resource Description Framework to describing networks by way of the Network Description Language. By using the linking capabilities of RDF, a distributed Topology Knowledge Base as described by Travostino [p41] can easily be created.

NDL unlocks the potential of machine-readable metadata about the network for control-planes, service planes and other applications that require data about the network. The goal of NDL is to provide a common topology information base and thus facilitate interworking between networks.

By leveraging the `seeAlso` property in RDF, it is possible to create a distributed (decentral) information system to describe networks.

The use of semantic web solves the addressing problem by providing URIs as globally unique addresses.

The investment in codifying models and RDF vocabularies pays dividend when new tools emerge that harvest upon these models, and unite network descriptions across administrative boundaries. By building upon existing Semantic Web techniques, applications are now appearing.